CE-105L Computer Programming for Civil Engineers – Matlab
Symbolic, Numerical Calculus and Differential Equation and Linear Algebra
Engr. Faisal ur Rehman

Notes Avaiable on:
enggprog.com (Engineering Programs)

Symbolic

class(E): Returns the class of expression. Or return class name of object.
If double, expression is number and if sym then expression is symbolic. e.g.,

```
>> name = class(pi)
name =
double

>> syms x
>> class(x+2)
ans =
sym

class returing handle
>> h = @sin;
class(h)
ans =
function_handle
```

digits(d): DIGITS Set variable precision digits. Digits determines the accuracy of variable precision (vpa) numeric computations. D = DIGITS returns the current setting of Digits. D is an integer. Default value is 32. After setting digits, calculation is then performed with vpa. e.g.,

```
>> syms x
>> E=(x+3)^.2;
>> old = digits;
>> digits(6)

substituting value of x with 0.2 in expression E using subs function and applying vpa.
>> vpa(subs(E,.2))
ans =
1.26191
```

```
>> vpa(pi)
ans =
3.14159
```

Other way of using vpa is as second argument of vpa.

```
>> vpa(pi,3)
ans =
3.14

>> vpa(pi,7)
ans =
3.141593
```

ezplot(E): Easy-to-use function plotter. e.g.,

```
>> syms x y
>> E1 = x^2+4;
```

ezplot with given x-values

```
>> ezplot(E1, [3 10])
```

ezplot with default x-values i.e., -2pi to 2pi

```
>> ezplot(E1)

>> E2 = x^2-y^2+4;
```

ezplot with given x and y values

```
>> ezplot(E2, [-3 4 -4 6])
```

ezplot with default x and y values i.e., -2pi to 2pi

```
>> ezplot(E2)
```

findsym(E): Finds the symbolic in given expression. If number n is specified as second argument then gives n number of symbolic variables near to x.

```
>> syms x y
>> a = 7;
>> E = (x+y)^a;
>> findsym(E)
ans =
```

```
x,y
>> findsym(E,1)
ans =
x
```

[N,D] = numden(A): numden- Numerator and denominator of a symbolic expression.

```
>> [n,d] = numden(sym(4/5))
n =
4
d =
5

>> syms x y
>> [n,d] = numden(x/y + y/x)
n =
x^2 + y^2
d =
x*y
```

sym and syms: Creates symbolic variables

To create real number variable x.

```
>> x=sym('x','real')
x =
x
```

or

```
>> x=sym('x')
x =
x
```

To create unreal (imaginary) number variable y

```
>> y=sym('y','unreal')
y =
y
```

To create real variable p, q, r and s in single command

```
>> syms p q r s 'real'
```

or

```
>> syms p q r s
```

collect(E): Collects coefficients of same power and same variable in an expression. e.g.,

```
>> syms x y
>>  collect(x^2*y + y*x - x^2 - 2*x)
ans =
(y - 1)*x^2 + (y – 2)*x
```

to collect with respect to (w.r.t) y variable:

```
>> collect(x^2*y + y*x - x^2 - 2*x,y)
ans =
(x^2 + x)*y - x^2 – 2*x
```

expand(E): Expands the given expression by carrying out power. e.g.,

```
>> syms x y
>> expand((x + y)^2)
ans =
x^2 + 2*x*y + y^2
```

factor(E): Factorize the expression. e.g.,

```
>> syms x
>> factor(x^2+2*x+1)
ans =
(x + 1)^2
```

poly2sym(matrix): Converts polynomial (matrix or vector) to symbolic expression. e.g.,

```
>> poly2sym([2 3 4 0 6])
ans =
2*x^4 + 3*x^3 + 4*x^2 + 6
```

sym2poly(E): Converts the symbolic expression to polynomial or matix. e.g.,

```
>> sym2poly(2*x^4 + 3*x^3 + 4*x^2 + 6)
ans =
    2     3     4     0     6
```

pretty(E): Writes expression in mathematical form as we write mathematical symbols by hand.

```
>> syms x
>> E = sqrt(x +2)/(3*x^2+x+3);
>> pretty(E)
```

```
        1/2
  (x + 2)
 ------------
    2
 3 x  + x + 3
```

[r how] = simple(E): Searches for the simplest form of an expression or matrix. It can also display result and the method by which it use simplification. It uses collect, expand and simplify command for simplification. e.g.,

```
>> syms x
>> E1=cos(x)^2+sin(x)^2;
>> [r how] = simple(E1)

r =

1


how =

simplify

>> E2=2*cos(x)^2-sin(x)^2;
>> [r how] = simple(E2)

r =

2 - 3*sin(x)^2


how =

simplify
```

simplify(E): Simplify symbolic expression using maple simplification method. e.g.,

```
>> syms x
>> simplify(sin(x)^2 + cos(x)^2)

ans =

1
```

subs(E, old, new): Substitute sumbolic expression from old with new variable, or value. It can be used to define a function. e.g.,

```
 >> syms x y
>> E1 = (x+2)^x;
>> E2 = subs(E1,x,y)

E2 =

(y + 2)^y

>> syms z
>> E3 = x+y+3*x^2;
>> E4 = subs(E3,y,z)

E4 =

3*x^2 + x + z

>> E5=subs(E1,x,0.5)

E5 =

   1.581138830084190

>> syms t
>> f=sym('f(t)');
>> g=subs(f,t,t-2)-f

g =

f(t - 2) – f(t)
```

solve(E): Solves symbolic equations. e.g.,

```
>>solve('x^3+3*x^2+3*x+3=0')

ans =

                          (-2)^(1/3) - 1
 - (-2)^(1/3)/2 - 1 - ((-2)^(1/3)*3^(1/2)*i)/2
     –    (-2)^(1/3)/2 - 1 + ((-2)^(1/3)*3^(1/2)*i)/2

>> eq1='p+q=9';
```

```
>> eq2='p+3*p=-5';
>> [p q]=solve(eq1,eq2)

p =

-5/4


q =

41/4
```

diff(E,v,n): Find the differential of expression E w.r.t given variable v with given order of n.

```
>> syms x y
>> E1 = x^3+x^2*y+y^2*x+3*x*y;
>> diff(E1,x,2)

ans =

6*x + 2*y
```

int(E, v, a,b): Performs the integral for given variable v. Also calculates the definite integral if limits a and b are provided. e.g.,

```
>> syms x y
>> E1 = x^3+x^2*y+y^2*x+3*x*y;
>> int(E1,y)

ans =

(x*y*(6*x^2 + 3*x*y + 2*y^2 + 9*y))/6

>> E2=sin(x)*log(x);
>> int(E2,x,2,10)

ans =

cosint(10) - cosint(2) + cos(2)*log(2) - cos(10)*log(10)

>> double(ans)

ans =
```

1.175145326576247

limit(E, v, a, 'd'): Performs the limit on the given expression w.r.t variable v to value of a and in specified direction d which can be left or right. e.g.,

```
>> limit('1/x',x,0,'right')

ans =

Inf

>> limit('1/x',x,0,'left')

ans =

-Inf
```

symsum(E): Performs symbolic summation of series for given expression. e.g.,

```
>> syms k
>> E=(k+1)^2;
>> symsum(E,1,5)
ans =

90
```

taylor(f,n,a): Calculates the taylor series for n-1 terms and for given value of variable x equal to a. e.g.,

```
>> syms x
>> f = 1/(5 + 4*cos(x));
>> T = taylor(f, 8)

T =

(49*x^6)/131220 + (5*x^4)/1458 + (2*x^2)/81 + 1/9

>> subs(T,x,3)

ans =

   0.883333333333333
```

Numerical Calculus and Differential Equations

diff(Matrix): Calculates the backward difference of given matrix. e.g.,

```
>> diff([1 2 5 5 3])

ans =

    1     3     0    -2
```

trapz(x,y): Calculates area under curve for given points as matrix of x and y. e.g.,

```
>> x=[1:.05:5];
>> y=sin(x);
>> A1=trapz(x,y)

A1 =

   0.256586651485251
```

quad('func',a,b): Calculates the area under curve using simpson rule. e.g.,

```
>> A2=quad('sin',1,5)

A2 =

   0.256640111828176
```

quadl('func',a,b): Calculates the area under curve using lobatto quarature. e.g.,

```
>> A3=quadl('sin',1,5)

A3 =

   0.256640077271133
```

linspace(x1,x2,n): Create a vector of n linearly spaced numbers from x1 to x2. e.g.,

```
>> linspace(1,5,5)
ans =

    1     2     3     4     5

>> linspace(1,5,3)

ans =

    1     3     5
```

Linear Algebra

det(A): Determines the determinant of a matrix A. e.g.,

```
>> A=[2 3 4;3 24 5; 15 16 7];
>> det(A)

ans =

  -910
```

Crammer Rule: X = inv(A) * B or X = A\B

rank(A): Calculates the rank of matrix A.

```
>> A=[2 3 4;3 24 5; 15 16 7];
>> rank(A)

ans =

     3
```

rref(A): Calculates the Reduced row echelon form of matrix A. e.g.,

```
>> A = magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> R = rref(A)

R =

     1     0     0     1
     0     1     0     3
     0     0     1    -3
     0     0     0     0
```

pinv(A): Calculates Moore-Penrose pseudoinverse of matrix.

```
>> A = magic(8)
```

```
A =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1

>> A = A(:,1:6) % makes 8 by 6 of matrix A that happens to be rank(A) = 3

A =

    64     2     3    61    60     6
     9    55    54    12    13    51
    17    47    46    20    21    43
    40    26    27    37    36    30
    32    34    35    29    28    38
    41    23    22    44    45    19
    49    15    14    52    53    11
     8    58    59     5     4    62

>> b = 260*ones(8,1)

b =

   260
   260
   260
   260
   260
   260
   260
   260

>> x = pinv(A)*b

x =

   1.153846153846155
   1.461538461538460
   1.384615384615385
```

1.384615384615383
1.461538461538459
1.153846153846152

norm(A): Calculates the norm of matrix. e.g.,
using above example,
>> norm(x)

ans =

3.281650616569466